Thunderhead Beowulf Cluster Workshop

November 22, 2004

User support

Jelena Marshak

CODE 587 NASA/GSFC

Introduction

- **Serving user in code optimization**
  - Going from serial to parallel
  - Checkpointing
  - Other issues?
- **Providing support in the form of**
  - Coaching
  - Classes
  - Consultation

Parallel programming

- Programmability, performance and portability
- Parallel Computational Models
- Introduction to OpenMP, MPI and SMS
- Overview: attractive features and limitations

Parallel programming
Programmability, performance, portability

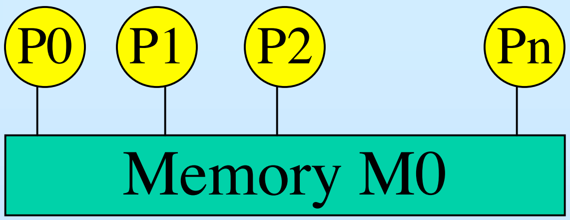
- **Programmability:** *Minimization* of “pain to gain ratio”:
  - modifications to serial code
  - parallelization effort
- **Performance:** *Measure* of how well the power of parallel processors is utilized to solve the problem.
- **Portability:** “*Measure* of the cost of porting, compared to the cost of redevelopment.” (Jim Mooney, West Virginia University)

  An application is considered portable if the effort required to
  - transport and
  - adapt

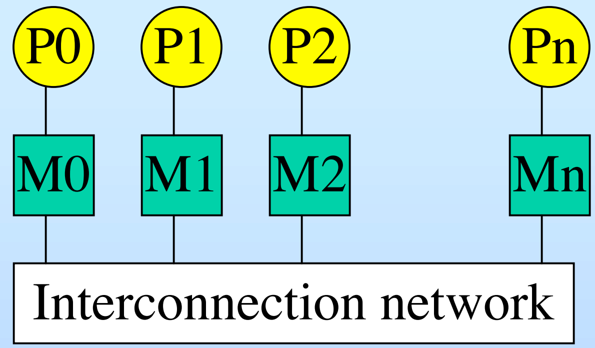
  it to a new environment is less than the effort of redevelopment.

Parallel programming
Computational Models

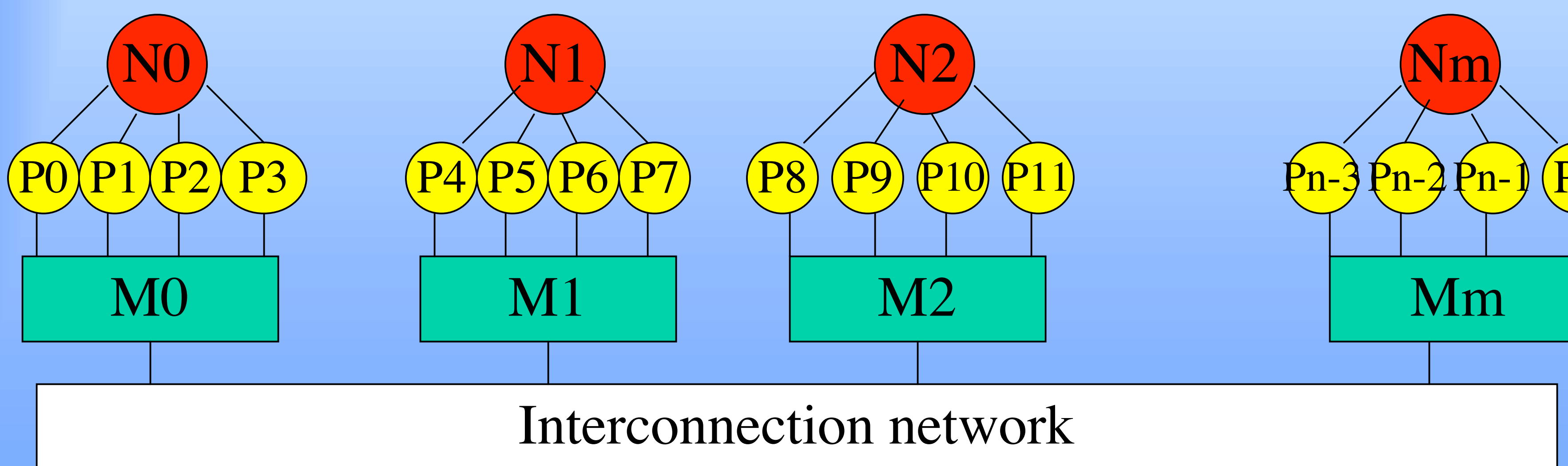
Shared memory model (SV1)



The message passing model (T3E)



The cluster model (HP/Compaq)



Parallel programming
What is OpenMP?

- **OpenMP** - Parallel Programming Model
- Industry standard (1997)
- Comprises of a set of compiler directives, along with a supporting library
- Converts Fortran, C/C++ code and directives into a parallel version
- Designed for shared memory machines

Procedure

1. f90 parallel_code.f -omp or kf90 -fkapargs='-conc' serial_code.f
2. a.out

Parallel programming
What is MPI?

- **MPI - M**essage **P**assing **I**nterface
- Standardized (1994) library of routines for parallel computers
- The basic communication mechanism of MPI is the transmission of data between a pair of processors
- Converts Fortran, C, or C++ code into a parallel version
- Links MPI library
- Designed for distributed memory machines

Procedure

1. f90 parallel_code.f -lmpi
2. prun -N8 -n32 a.out

Parallel programming
What is SMS?

- **SMS - S**calable **M**odel **S**ystem
- Directive-based software designed at NOAA Forecast Systems Laboratory
- Translates the Fortran code and directives into a parallel version
- Links MPI and SMS libraries
- Designed for shared and distributed memory machines

Procedure:

1. setenv SMS /usr/ulocal/sms-2.7.0.r8i8
2. $(SMS)/bin/ppp parallel_code.f
3. F90 -I$(SMS)/include -L$(SMS)/lib -lsms -lmpi parallel_code_sms.f
4. $(SMS)/bin/smsRun -cf config.64 a.out

Parallel programming

OpenMP, MPI and SMS examples

MPI

```
c only for pe's on bottom edge
      if (pe_j_p(1) .lt. 0) then
         ABER=0.
         do 668 i=1,im
         ACR=0.25*(DX(I,2)+DX(I,1))*FSM(I,1,1)*
    &             (H(I,1)+H(I,2)+ELF(I,1)+ELF(I,2))
         ABER=ABER+ACR
 668     CONTINUE
```

OpenMP

```
      ABER=0.
!$OMP PARALLEL DO DEFAULT(SHARED) PRIVATE(ACR,i) REDUCTION(+:ABER)
      DO 668 I=1,IM
      ACR=0.25*(DX(I,2)+DX(I,1))*FSM(I,1)*
     &          (H(I,1)+H(I,2)+ELF(I,1)+ELF(I,2))
      ABER=ABER+ACR
  668 CONTINUE
!$OMP END PARALLEL DO
```

SMS

```
      ABER=0.
CSMS$reduce(aber,SUM) BEGIN
      DO I=1,IM
CSMS$global_index(<decomp:2>,<decompA:2>) begin
      ABER=ABER+0.25*(DX(I,2)+DX(I,1))*FSM(I,1,1)*
     &                (H(I,1)+H(I,2)+ELF(I,1)+ELF(I,2))
CSMS$global_index end
      ENDDO
CSMS$reduce end
```

Parallel programming
Overview

Technique / Feature	Open MP	MPI	SMS
Integrity of the code	+		+
Performance		+	+
Incremental parallelization	+		+
Portability	+	+	+
Dependency handling	+		+
Nesting and coupling			+
Use of run time environment variables	+	+	+
Debugging			+

Checkpointing

- **Checkpointing routines**
  - Provide routines for checkpointing runs:
    - one that writes a checkpoint file, and
    - another that can read it back in to restart the run (periodic/on interruption).
- **Checkpointing packages**
  - Research among existing packages (EPCKPT, CRAK, Dynamic, Porch and etc.)

Checkpointing

Checkpointing packages

- EPCKPT Parallel processes, shared memory, semaphores, etc. (Linux 2.4.2.)
- CRAK A module based on EPCKPT. Same features. (Linux 2.4.x.)
- SBUML Checkpointing, compression, and migration of user-mode Linux virtual machines. No kernel modifications required. (Linux 2.4.23.)
- Checkpointing Research Team Both kernel and user-level checkpointing. (Solaris 8/9.)
- Asim Shankar's Project Another Linux user-level checkpointing library (Linux 2.4.x.)
- Dragon Fly BSD BSD kernel distribution. (BSD.)
- Dynamite User-level. No recompilations nor relinking. Includes support for parallel processes (PVM/MPI). (Linux and Solaris.)
- Porch Portable across heterogeneous machines. User-level compiler infrastructure. (Linux, AIX, Solaris, HPUX, Irix, FreeBSD.)
- Esky User-level checkpointing. Works under Linux 2.2 and Solaris 2.6 and is written to be independent of CPU type. (Linux/Solaris.)
- CKPT User-level checkpointing library for Linux. Does not require recompilation nor relinking of applications. (Linux.)
- Scalable Systems Checkpoint Kernel-level checkpoint system for MPI applications. (Linux.) Source NA
- Chpox Kernel module. (Linux.)
- Condor User level checkpointing and other tools for batch job scheduling.(UNIX 6.3, NT 6.2)

References

1. Blumberg, A. F., and G. L. Mellor, *A description of a three dimensional coastal ocean circulation model*, in Three-Dimensional Coastal Ocean Models, Coastal Estuarine Sci.Ser.,vol.4, edited by N. S. Heaps, pp. 1-16, AGU, Washington, D.C., 1987.
2. Marc Snir, Steve Otto, Steven Huss Lederman, David Walker, Jack Dongara, *MPI-The Complete Reference*, The MIT Press., Cambridge, Massachusetts, 1998.
3. W. Oberpriller, A. C. Sawdey, M. T. O'Keefe, S. Gao, and S. A. Piacsek, *Parallelizing the Princeton Ocean Model Using TOPAZ*, http://www.borg.umn.edu/topaz/mppom/
4. Rohit Chandra, Leonardo Dagum, Dave Kohr, Dror Maydan, Jeff McDonald,Ramesh menon, *Parallel Programming in OpenMP*, 2001
5. M.W.Govett, D.S.Schaffer, T.Henderson, L.B.Hart, J.P.Edwards, C.S.Liou, T-L.Lee, *SMS: A Directive-Based Parallelization Approach for Shared and Distributed High Performance Computers*, http://www-ad.fsl.noaa.gov/ac/sms.html
6. http://www.checkpointing.org

Thunderhead Beowulf Cluster Workshop

November 22, 2004

User support

Jelena Marshak

CODE 587 NASA/GSFC

SMS Restrictions

Open-Source Software License/Disclaimer

- Forecast Systems Laboratory
- NOAA/OAR/FSL
- 325 Broadway Boulder, CO 80305
- (adopted November 2000)

- This software is distributed under the Open Source Definition, which may be found at http://www.opensource.org/osd.html.

- In particular, redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- * Redistributions of source code must retain this notice, this list of conditions and the following disclaimer.
- * Redistributions in binary form must provide access to this notice, this list of conditions and the following disclaimer, and the underlying source code.
- * All modifications to this software must be clearly documented, and are solely the responsibility of the agent making the modifications.
- * If significant modifications or enhancements are made to this software, the FSL Software Policy Manager (softwaremgr@fsl.noaa.gov) should be notified.

- THIS SOFTWARE AND ITS DOCUMENTATION ARE IN THE PUBLIC DOMAIN AND ARE FURNISHED "AS IS." THE AUTHORS, THE UNITED STATES GOVERNMENT, ITS INSTRUMENTALITIES, OFFICERS, EMPLOYEES, AND AGENTS MAKE NO WARRANTY, EXPRESS OR IMPLIED, AS TO THE USEFULNESS OF THE SOFTWARE AND DOCUMENTATION FOR ANY PURPOSE. THEY ASSUME NO RESPONSIBILITY (1) FOR THE USE OF THE SOFTWARE AND DOCUMENTATION; OR (2) TO PROVIDE TECHNICAL SUPPORT TO USERS

Ice and Ocean Model

MODEL

start

initialize

begin calculation

get forcing data

execute Ice Model

update Ice area

execute Ocean Model

output restart data

end

November 22, 2004

3 -level snow-ice system

snow	snow
ice	ice
ice	ice

ICE

compute: ice thermodynamical variables
advection and diffusion
internal ice stress
ice dynamics
output: monthly averages

OCEAN

compute:
advection and diffusion of momentum
baroclinic pressure gradient

do short time step calculation

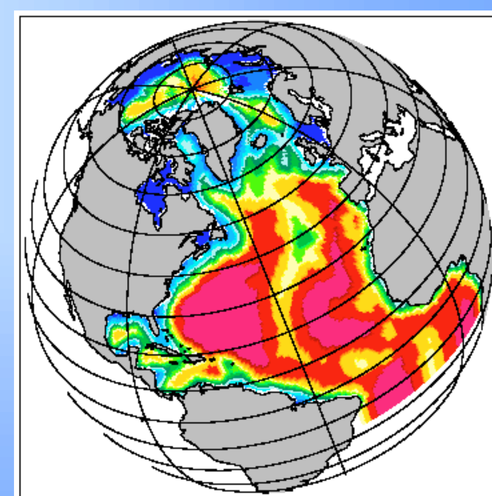
compute:
surface elevation
vert. mean velocities

adjust:
vertically averaged velocities
compute:
horizontal and vertical velocities
turbulence quantities
advection and diffusion of scalars
output:
monthly averages

Princeton Ocean Model is a sigma coordinate, free surface, primitive equation ocean model, which includes a turbulence sub-model.

It was developed in the late 1970's by Blumberg and Mellor. During 1986 - 1990 Arctic Ocean Model, developed by Kantha and Hakkinen was coupled to the Ocean Model.

The coupled model has been used for modeling of Arctic - Atlantic oceans.



Study area

Notes

* Big-endian or little-endian byte order

This issue refers to which bytes are most significant in multi-byte data types. In big-endian architectures, the leftmost bytes (those with a lower address) are most significant. In little-endian architectures, the rightmost bytes are most significant. The terms big-endian and little-endian are derived from the Lilliputians of Gulliver's Travels, whose major political issue was whether soft-boiled eggs should be opened on the big side or the little side. Our issue here is that the HP/Compaq SC45 using Alpha chips becomes the first little-endian Super Computer installed at NCCS.

OpenMP code example

Parallel code

```
      SUBROUTINE ADVAVE(ADVUA,ADVVA,MODE)
      INCLUDE 'comblk.o'
      DIMENSION ADVUA(IM,JM),ADVVA(IM,JM),CURV2D(IM,JM)
      EQUIVALENCE (TPS,CURV2D)
!$OMP PARALLEL DEFAULT(SHARED) PRIVATE(I,J)
!$OMP DO
        DO 300  J=2,JM
        DO 300  I=2,IMM1
  300   FLUXUA(I,J)=.125*((D(I+1,J)+D(I,J))*UA(I+1,J)
     &              +(D(I,J)+D(I-1,J))*UA(I,J))*(UA(I+1,J)+UA(I,J))
!$OMP END DO nowait
```

Serial code

```
      SUBROUTINE ADVAVE(ADVUA,ADVVA,MODE)
      implicit none
      include 'comblk.g'
      DIMENSION ADVUA(IM,JM),ADVVA(IM,JM),CURV2D(IM,JM)
      EQUIVALENCE (TPS,CURV2D)
      DO 300 J=2,JM
      DO 300 I=1,IMM1
 300  FLUXUA(I,J)=.125*((D(I+1,J)+D(I,J))*UA(I+1,J)
     1           +(D(I,J)+D(I-1,J))*UA(I,J))*(UA(I+1,J)+UA(I,J))
```

MPI code example

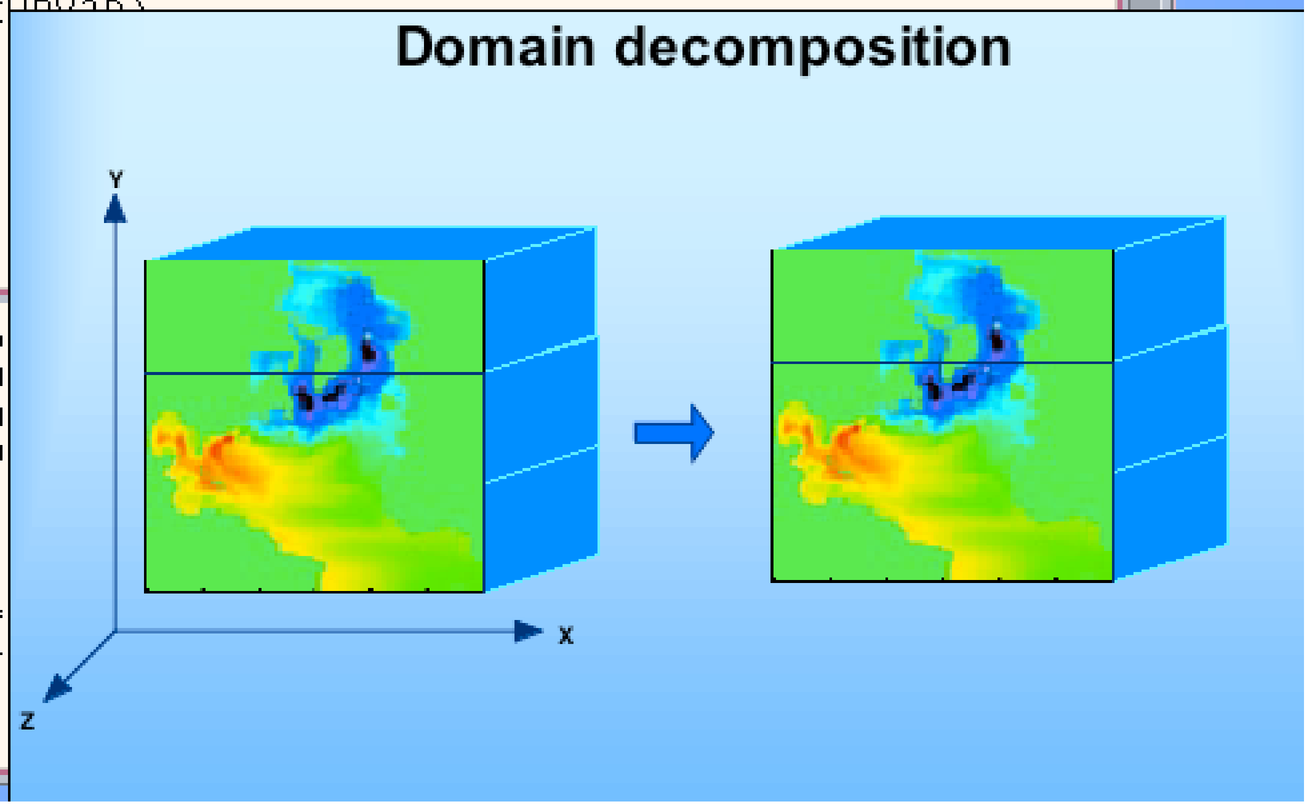
Terminal

Window Edit Options Help

```
      SUBROUTINE ADVAVE(ADVUA,ADVVA,MODE)
      IMPLICIT NONE
      INCLUDE 'oidimensions.h'
      INCLUDE 'oicomblkg.h'
      REAL ADVUA(-NBDY+1:IDM+NBDY,-NBDY+1:JDM+NBDY),
     &     ADVVA(-NBDY+1:IDM+NBDY,-NBDY+1:JDM+NBDY),
     &     CURV2D(-NBDY+1:IDM+NBDY,-NBDY+1:JDM+NBDY)
      EQUIVALENCE (TPS,C
      do 300 J=jS2P1,JM
      do 300 I=iS2P0,iSi
 300  FLUXUA(I,J)=.125*(
     1            +(D(I,J
```

Parallel code

```
      SUBROUTINE A
      implicit non
      include 'com
      DIMENSION AD
      EQUIVALENCE
      DO 300 J=2,J
      DO 300 I=1,I
 300  FLUXUA(I,J)=
     1            +
```



MPI code example

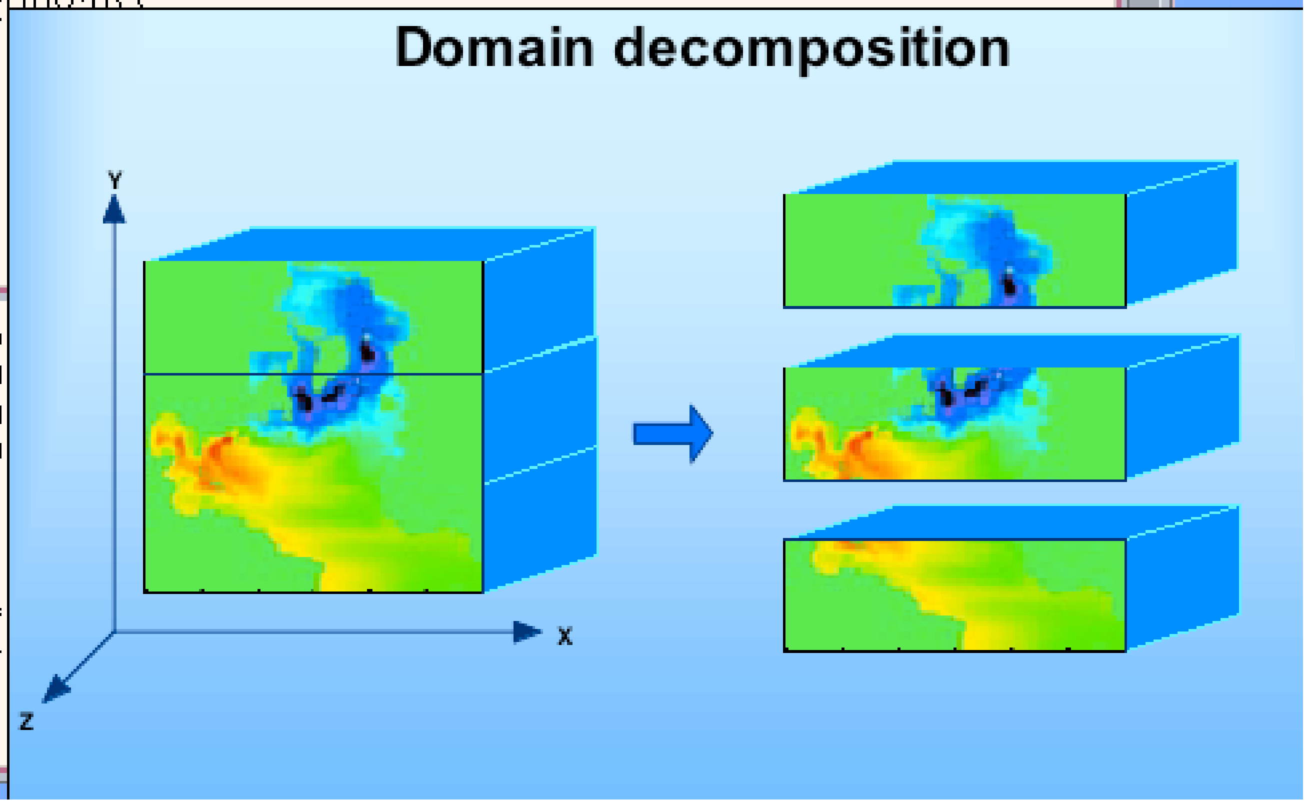
Terminal

Window Edit Options Help

```
      SUBROUTINE ADVAVE(ADVUA,ADVVA,MODE)
      IMPLICIT NONE
      INCLUDE 'oidimensions.h'
      INCLUDE 'oicomblkg.h'
      REAL ADVUA(-NBDY+1:IDM+NBDY,-NBDY+1:JDM+NBDY),
     &     ADVVA(-NBDY+1:IDM+NBDY,-NBDY+1:JDM+NBDY),
     &     CURV2D(-NBDY+1:IDM+NBDY,-NBDY+1:JDM+NBDY)
      EQUIVALENCE (TPS,C
      do 300 J=jS2P1,JM
      do 300 I=iS2P0,iSi
 300  FLUXUA(I,J)=.125*(
     1           +(D(I,J
```

Parallel code

```
      SUBROUTINE A
      implicit non
      include 'com
      DIMENSION AD
      EQUIVALENCE
      DO 300 J=2,J
      DO 300 I=1,I
 300  FLUXUA(I,J)=
     1           +
```



MPI code example

Terminal

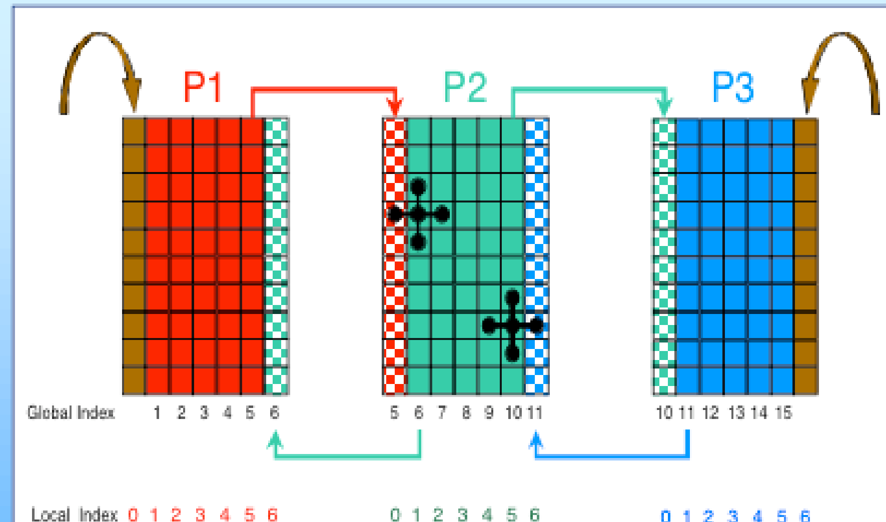
Window Edit Options Help

```
      SUBROUTINE ADVAVE(ADVUA,ADVVA,MODE)
      IMPLICIT NONE
      INCLUDE 'oidimensions.h'
      INCLUDE 'oicomblkg.h'
      REAL ADVUA(-NBDY+1:IDM+NBDY,-NBDY+1:JDM+NBDY),
     &     ADVVA(-NBDY+1:IDM+NBDY,-NBDY
     &     CURV2D(-NBDY+1:IDM+NBDY,-NBD
      EQUIVALENCE (TPS,CURV2D)
      do 300 J=jS2P1,JM
      do 300 I=iS2P0,iSimm1Pim
  300 FLUXUA(I,J)=.125*((D(I+1,J)+D(I,J
     1           +(D(I,J)+D(I-1,J))*UA(
```

Parallel code

```
      SUBROUTINE ADVAVE(ADVUA,ADV
      implicit none
      include 'comblk.g'
      DIMENSION ADVUA(IM,JM),ADVV
      EQUIVALENCE (TPS,CURV2D)
      DO 300 J=2,JM
      DO 300 I=1,IMM1
  300 FLUXUA(I,J)=.125*((D(I+1,J)+D(I,J))*UA(I+1,J)
     1           +(D(I,J)+D(I-1,J))*UA(I,J))*(UA(I+1,J)+UA(I,J))
```

Adjacent data dependency



Courtesy of Dan Schaffer (NOAA)

SMS code example

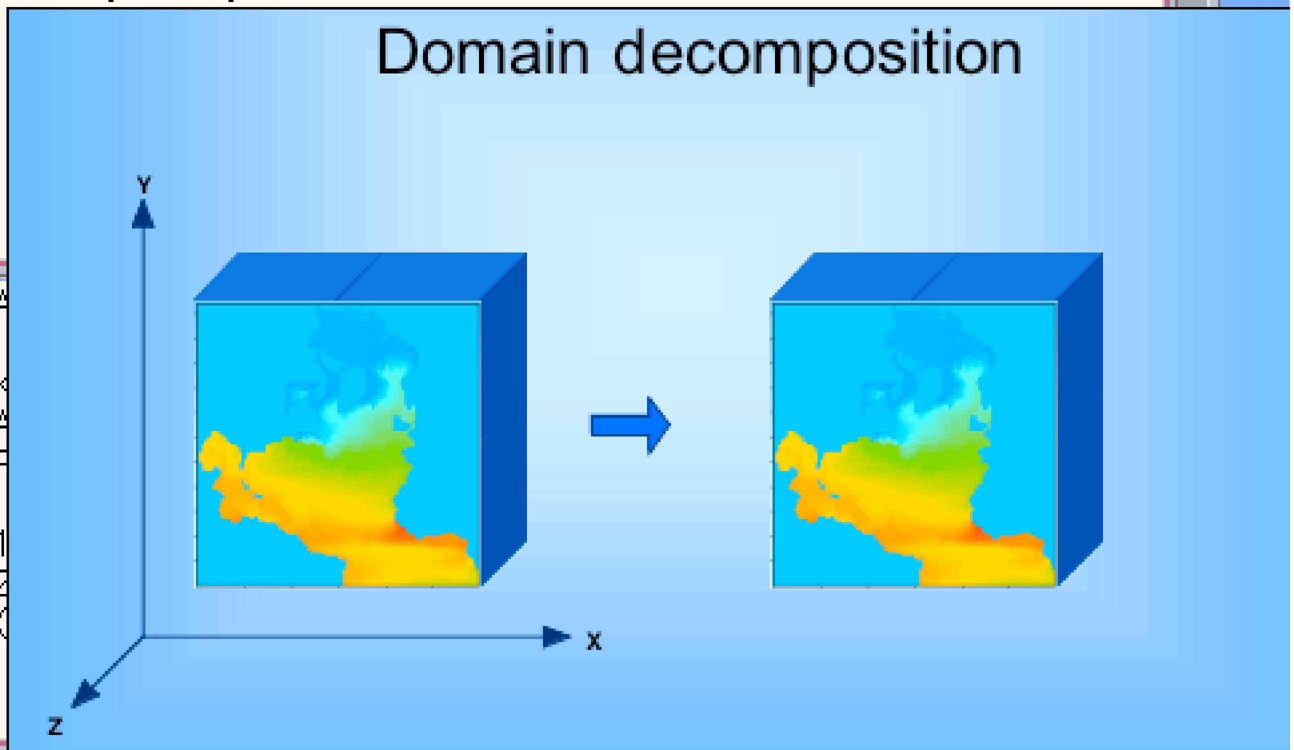
Terminal

Window Edit Options Help

```
      SUBROUTINE ADVAVE(ADVUA,ADVVA,MODE)
      INCLUDE 'comblk.g'
CSMS$DISTRIBUTE (decomp,<im>,<jm>) BEGIN
      DIMENSION ADVUA(IM,JM),ADVVA(IM,JM),CURV2D(IM,JM)
CSMS$DISTRIBUTE END
      EQUIVALENCE (TPS,CURV2D)
CSMS$CHECK_HALO(ua<0,1><1,0>,va<1,0><0,1>,vab<1,0><0,1>,uab<0,1><1,0>,
CSMS$> d<1,1><1,1>,aam2d<1,0><1,0>)
CSMS$PARALLEL (decomp,
      DO 300 J=2,JM
      DO 300 I=2,IMM1
 300  FLUXUA(I,J)=.125
     1           +(D(I
```

Parallel code

```
      SUBROUTINE ADVA
      implicit none
      include 'comblk
      DIMENSION ADVUA
      EQUIVALENCE (TP
      DO 300 J=2,JM
      DO 300 I=1,IMM1
 300  FLUXUA(I,J)=.12
     1           +(D(
```



SMS code example

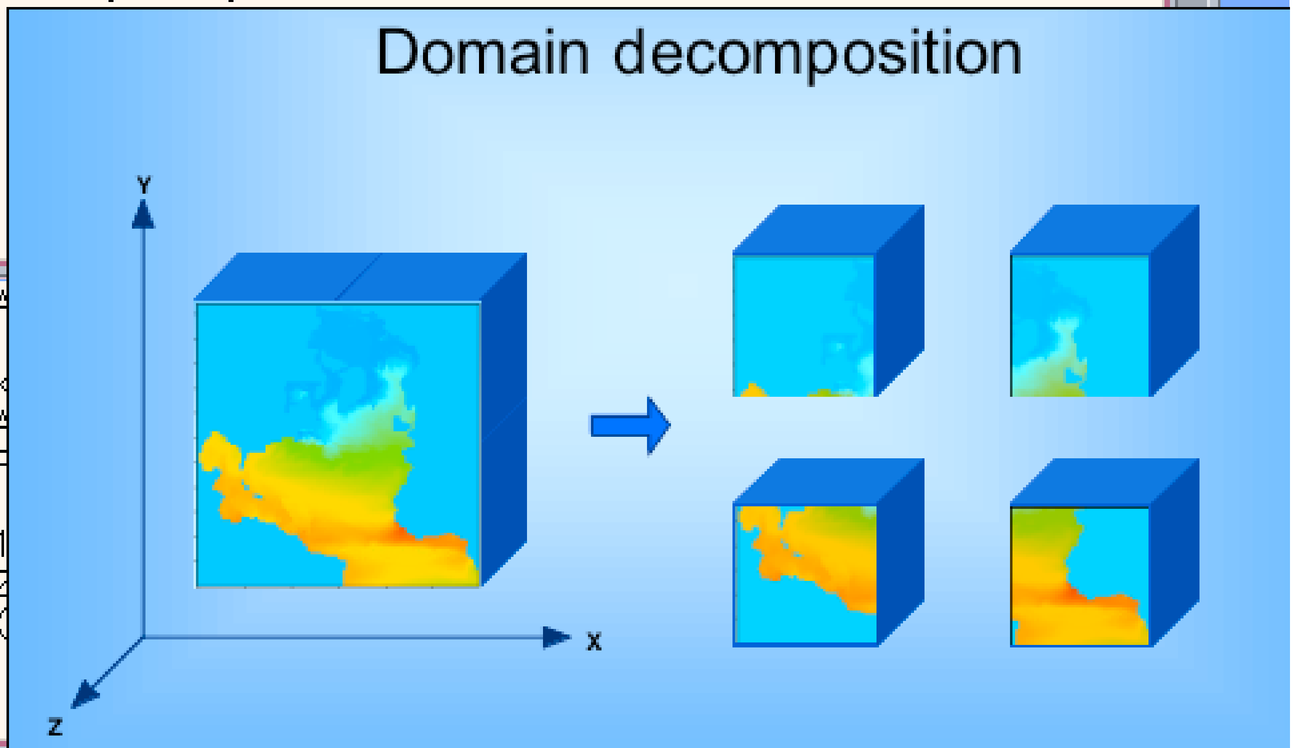
Terminal

Window Edit Options Help

```
      SUBROUTINE ADVAVE(ADVUA,ADVVA,MODE)
      INCLUDE 'comblk.g'
CSMS$DISTRIBUTE (decomp,<im>,<jm>) BEGIN
      DIMENSION ADVUA(IM,JM),ADVVA(IM,JM),CURV2D(IM,JM)
CSMS$DISTRIBUTE END
      EQUIVALENCE (TPS,CURV2D)
CSMS$CHECK_HALO(ua<0,1><1,0>,va<1,0><0,1>,vab<1,0><0,1>,uab<0,1><1,0>,
CSMS$> d<1,1><1,1>,aam2d<1,0><1,0>)
CSMS$PARALLEL (decomp,
      DO 300 J=2,JM
      DO 300 I=2,IMM1
 300  FLUXUA(I,J)=.125
     1          +(D(I
```

Parallel code

```
      SUBROUTINE ADVA
      implicit none
      include 'comblk
      DIMENSION ADVUA
      EQUIVALENCE (TF
      DO 300 J=2,JM
      DO 300 I=1,IMM1
 300  FLUXUA(I,J)=.12
     1          +(D(
```



SMS code example

Terminal

Window Edit Options Help

Parallel code

```
      SUBROUTINE ADVAVE(ADVUA,ADVVA,MODE)
      INCLUDE 'comblk.g'
CSMS$DISTRIBUTE (decomp,<im>,<jm>) BEGIN
      DIMENSION ADVUA(IM,JM),ADVVA(IM,JM),CURV2D(IM,JM)
CSMS$DISTRIBUTE END
      EQUIVALENCE (TPS,CURV2D)
CSMS$CHECK_HALO(ua<0,1><1,0>,va<1,0><0,1>,vab<1,0><0,1>,uab<0,1><1,0>,
CSMS$> d<1,1><1,1>,aam2d<1,0><1,0>)
CSMS$PARALLEL (decomp,<i>,<j>) BEGIN
      DO 300 J=2,JM
      DO 300 I=2,IMM1
 300  FLUXUA(I,J)=.125*((D(I+1,J)+D(I,J))*UA(I+1,J)
     1           +(D(I,J)+D(I-1,J))*UA(I,J))*(UA(I,J)+UA(I,J-1))
```

Serial code

```
      SUBROUTINE ADVAVE(ADVUA,ADVVA,MODE)
      implicit none
      include 'comblk.g'
      DIMENSION ADVUA(IM,JM),ADVVA(IM,JM),CURV2D(IM,JM)
      EQUIVALENCE (TPS,CURV2D)
      DO 300 J=2,JM
      DO 300 I=1,IMM1
 300  FLUXUA(I,J)=.125*((D(I+1,J)+D(I,J))*UA(I+1,J)
     1           +(D(I,J)+D(I-1,J))*UA(I,J))*(UA(I+1,J)+UA(I,J))
```

Parallel programming
Application to the Ice-Ocean Model

General porting issues

- Data format
    - Little-endian or Big-endian byte order
    - Length of Real and Integer
- Libraries

Solutions

- Use of F90 -i8 -r8 -convert big_endian
- Use of Real() function
- Use of double precision functions
- Use of MPI_DOUBLE_PRECISION type in MPI functions

Results

Machine	Compaq	Cray SV1	Compaq	Cray T3E	Compaq	Cray T3E
Method	Serial	Serial	MPI	MPI	SMS	SMS
Number CPU-s	1	1	47	47	64	64
Time (sec)	3099.1	4298.1	142.8	1119.1	78.3	666.6

Parallel programming

Little-endian or Big-endian?

Compaq - Little-endian machine

SV1,T3E - Big-endian machines

Binary representation

SV1

63	62	61		48	47		0
signs		exponent			mantissa		

T3E

63	62	61		52	51		0
signs		exponent			mantissa		

Compaq

63	62		52	51		0
sign	exponent			mantissa		

Parallel programming
Kind/precision/range

KIND/precision/range for real values in NCCS Fortran 90 compilers

Computer	half precision*	Single Precision(default)	Double Precision	Quad Precision
Cray SV-1	4/6/2465	8/13/2465	16/28/2465	N/A
Cray T3E	4/6/37	8/15/307	N/A	N/A
SGI Origin 2K/3K	N/A	4/6/37	8/15/307	16/31/275
HP/Compaq SC45	N/A	4/6/37	8/15/307	16/33/4931

* The term half precision had once been used for Cray T3D system.

KIND/range for integer values in NCCS Fortran 90 compilers

Computer	int8	int16	int32	int64
Cray SV-1	1/2	2/4	4/9	8/18*
Cray T3E	1/2	2/4	4/9	8/18*
SGI Origin 2K/3K	1/2	2/4	4/9*	8/18
HP/Compaq SC45	1/2	2/4	4/9*	8/18

* system default

http://webserv.gsfc.nasa.gov/PRIVATE/NCCS/NCCSTAG/halem/porting.html

Parallel programming

Application to the Ice-Ocean Model

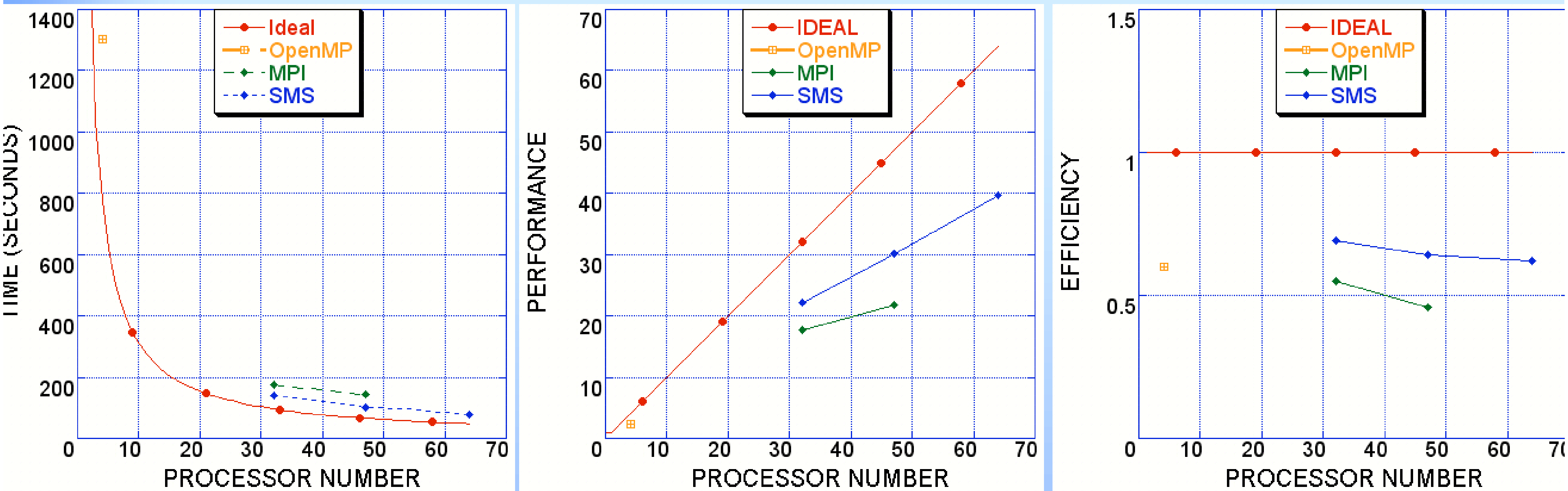
Method	Processes	Time (sec)	Ideal Time	Performance	Efficiency
Serial	1	3099.1	3099.1	1	NA
OpenMP	4	1301.6	774.8	2.4	0.6
MPI	32	175.1	96.8	17.7	0.55
MPI	47	142.8	65.9	21.7	0.46
SMS	32	139.8	96.8	22.2	0.69
SMS	47	102.5	65.9	30.2	0.64
SMS	64	78.3	48.4	39.6	0.62

0.93 vs. 0.83

0.69 vs. 0.55

25% faster

Parallel Programming
Application to the Ice-Ocean Model



The parallel code performance versus number of processors

Amdahl's law:

$$S(N) = \frac{1}{P_S + \frac{P_P}{N}}$$

N=4	N=47	N=64
P_S=0.22	P_S=0.02	P_S=0.01
P_P=0.78	P_P=0.98	P_P=0.99

Concluding remarks

NASA SMS version of POM coupled to Sea Ice Model has been in production since June, 2002. SMS is preferable to OpenMP and MPI for the following reasons:

Portable - Several multidecadal (1947-2001) numerical experiments were performed, after code was successfully ported to Compaq from T3E. Each experiment taking around a week of total time (7 times faster than on T3E)

Fast - SMS code performance is 25% better than the performance of the MPI code

Easy to maintain - The code is adaptable to changes without much of an effort